PRIFYSGOL ABERYSTWYTH UNIVERSITY

Aberystwyth University

Deterministic Parameter Control in Harmony Search

Diao, Ren; Shen, Qiang

Published in:
2010 UK Workshop on Computational Intelligence (UKCI)

DOI:
10.1109/UKCI.2010.5625576

Publication date:
2010

Citation for published version (APA):
Diao, R., & Shen, Q. (2010). Deterministic Parameter Control in Harmony Search. In *2010 UK Workshop on Computational Intelligence (UKCI)* https://doi.org/10.1109/UKCI.2010.5625576

General rights
Copyright and moral rights for the publications made accessible in the Aberystwyth Research Portal (the Institutional Repository) are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the Aberystwyth Research Portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the Aberystwyth Research Portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

tel: +44 1970 62 2400
email: is@aber.ac.uk

Download date: 05. May. 2023

Deterministic Parameter Control in Harmony Search

Ren Diao and Qiang Shen

Abstract— **Harmony search is a recently developed meta heuristic capable of solving discrete and continuous valued optimisation problems. However, the nature of pre-defined constant parameters limits the exploitation of the algorithm. This paper proposes a number of deterministic parameter control rules to fine-tune these parameters individually and dynamically, making Harmony Search a more dynamic algorithm which is able to achieve better results. A combined approach that implements all the proposed rules is then applied to various benchmarks and engineering problems. Experimental results reveal that the combined approach can find better solutions when compared to the original harmony search and several other heuristics, making harmony search a strong mechanism to perform optimisation tasks.**

Key words: Harmony Search; Meta Heuristics; Constrained Optimisation; Parameter Control

I. Introduction

Harmony Search (HS) is an evolutionary algorithm developed in [9]. As a meta-heuristic-based algorithm, conceptually, it mimics the improvisation process of music players. HS algorithm has been very successful in a wide variety of optimisation problems (e.g. [3], [17]), presenting several advantages over traditional optimisation techniques. In particular, it imposes fewer mathematical requirements and is not sensitive to the settings of initial parameter values. This makes its application implementation fairly straightforward. As the algorithm essentially performs stochastic random search, derivative information is also unnecessary.

Although it is also a population-based approach, the HS algorithm works by generating a new vector that encodes a candidate solution, after considering all of the existing vectors. This forms a sharp contrast with conventional evolutionary approaches such as genetic algorithms that consider only two (parent) vectors in order to produce a new vector. It increases the robustness and flexibility of the underlying search mechanism and hence, ensures better solutions. However, the nature of pre-defined constant parameters limits the exploitation of the algorithm.

Parameter tuning and parameter control are two different types of approach commonly used algorithm adaptation and search optimisation. The former means finding appropriate values for the parameters of an algorithm before running it, with such fine-tuned parameters fixed during the run. The latter starts with initial parameter settings which are then modified during the run. In particular, deterministic parameter control uses predefined parameter alteration strategies to modify parameters deterministically. Usually a time-dependent schedule is used, i.e. the modification rule will be used after a set number of iterations have elapsed since the last time the rule was activated. This is in contrast to adaptive parameter control, where feedback from the search is utilised to further determine the direction and/or magnitude of any changes made to the parameters. [8]

Recently, deterministic parameter control has been used to modify the pitch adjusting rate and bandwidth parameters of HS dynamically [14]. Although these modifications show promising results, the performance of the algorithm is still limited due to the constant setting of the remaining parameters. This paper proposes a number of deterministic rules to adjust HS parameters. It also suggests the use of scalable bandwidths in proportional to the value ranges for each variable. A combined approach that implements all the rules is also proposed, making HS a more dynamic and efficient algorithm that is capable of finding better solutions.

The rest of this paper is structured as follows. Section II introduces the main concepts of harmony search. Section III describes the parameter control rules for each individual parameters and the combined parameter adjustment approach. Section IV shows the experimentation carried out using the combined approach on various constrained and unconstrained optimisation problems, including real-world engineering designs, and discusses the experimental results. Section V concludes the paper and proposes further work in the area.

II. The Principles of Harmony Search

Harmony Search mimics the improvisation process of musicians, during which, each musician plays a note for finding a best harmony all together. When applied to optimisation problems, the musicians represent the decision variables of the cost function, and HS acts as a meta heuristic algorithm which attempts to find a solution vector that optimises this function. In the process, each decision variable (musician) generates a value (note) for finding a global optimum (best harmony). The Harmony Search algorithm has a novel stochastic derivative (for discrete variable) based on musician's experience, rather than gradient (for continuous variable) in differential calculus.

A. Key Concepts

The key concepts of HS algorithm are musicians, notes, harmonies and harmony memory. In most optimisation problems solvable by HS, the musicians are the decision variables of the function being optimised. The notes played by the musicians are the values each decision variable can take. The harmony contains the notes played by all musicians, or a solution vector containing the values for each decision attribute. The harmony memory contains harmonies played by the musicians, or a storage place for solution vectors.

Ren Diao and Qiang Shen are with the Department of Computer Science, Aberystwyth University, UK (email: {rrd09, qqs}@aber.ac.uk).

A more concrete representation of harmony memory is a two dimensional matrix, where the rows contain harmonies (solution vectors) and the number of rows are predefined and bounded by the harmony memory size. Each column is dedicated to one musician, and the entire column stores all the notes played by him in all harmonies, referred to as the note domain for each musician in this paper.

B. Iteration Steps and Algorithm Illustration

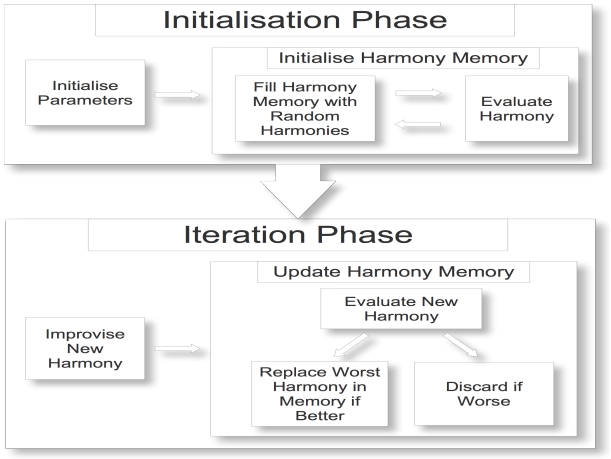

Fig. 1. Harmony Search Illustrated

Harmony Search can be divided into two core phases, initialisation and iteration, as shown in Fig. 1. A simple example problem taken from [9] is used for a better illustration: Minimise $(a-2)^2 + (b-3)^4 + (c-1)^2 + 3$ where a, b, c $\in$ {1,2,3,4,5}

1) Initialisation:

a) Initialise Problem Domain: The parameters of HS are assigned according to the problem, including: size of harmony memory, number of musicians, max iteration, and optionally, the harmony memory considering rate (HMCR), and the pitch adjustment rate (PAR). In the example, the number of musicians is 3, each corresponds to the decision attributes a, b and c. The harmony memory size is 3, and the objective function is the function to be minimised, the lower the better.

b) Initialise Harmony Memory: Harmony memory is filled with randomly generated solution vectors. In the example problem, 3 randomly generated solution vectors are {2,2,1}, {1,3,4} and {5,3,3}.

2) Iteration:

a) Improvise New Harmony: A new value is chosen randomly by each musician out of their note domain, and together forms a new harmony. In the example, musician a may randomly choose 1 out of {2,1,5}, b chooses 2 out of {2,3,3} and c chooses 3, forming a new harmony {1,2,3}.

There are two factors which affect the note choice of a musician, HMCR and PAR. HMCR, range from 0 to 1, is the rate of choosing one value from the historical notes stored in the harmony memory, while $(1 - HMCR)$ is the rate of randomly selecting one value from all possible range of values. If HMCR is set low, the musicians will constantly explore other areas of the solution space, and a higher HMCR will restrict the musicians to historical choices. The PAR parameter causes the musicians to select a neighbouring value based on the following formula $a \pm (random * BW)$, where BW is an arbitrary distance bandwidth, while $(1 - PAR)$ is the probability of using the chosen value without further alteration. The pitch adjustment is applied after a note is chosen by the musician, either from the harmony memory or all possible value range.

Given the above example, with $HMCR = 0.9$, and $PAR = 0.1$, musician a will choose out of all possible values with a probability of 10%. PAR will then force the musician to choose a neighbouring value with 10% probability, each of the two neighbours will then have equal chances being chosen. A wrap around approach may be taken for the edge values, 1 and 5, to ensure 2 neighbours for each value.

b) Update Harmony Memory: If the new harmony is better than the worst harmony in the harmony memory, judged by the objective function, the new harmony is then included in harmony memory and the existing worst harmony is removed. The new harmony {1,2,3} has the evaluation score of 9, making it better than the worst harmony in the memory {1,3,4} which has a score of 16, therefore the harmony {1,3,4} is removed from memory, replaced with {1,2,3}. If {1,2,3} had a larger score than 16, it would be the one being discarded.

The algorithm continues to iterate until the maximum number of iterations has been reached. In the example, if the musicians later choose {2,3,1}, which is very likely as those numbers are already in the note domains, the problem will be solved with a minimal score of 3.

C. A Probabilistic View of Harmony Search

In order to demonstrate the convergence capability of harmony search, consider the harmony memory with the following parameters: the size of the harmony memory (the number of harmonies in harmony memory) = M, the number of instruments (variables) = N, the number of possible notes (values) of an instrument = L, the number of optimal note (value) of instrument i in the harmony memory = H_i, $HMCR = H_r$, and the optimal harmony is (x,y,z). The probability of finding the optimal harmony is $Pr(H) = \subseteq_{i=1}^{N} \left[Hr \frac{H_i}{M} + (1 - Hr) \frac{1}{L} \right]$ [9] where the pitch adjusting rate is not considered because it is an optional operator.

Initially, the harmony memory is filled with random harmonies. If there is not any optimal note of all instruments in the harmony memory, $H_1 = H_2 = ... = H_N = 0$ and $Pr(H) = \left[(1 - Hr) \frac{1}{L} \right]$ This means that the probability $Pr(H)$ is very low. However, if the schema of optimal harmony such as (*,y,z), (x,*,z), (x,y,*) have better evaluation than others, the number of optimal notes of instrument i in the harmony memory, H_i will be increased iteration by iteration. Consequently, the probability of finding the optimal harmony, $Pr(H)$ will be increased.

III. Deterministic Parameter Control in HS

Traditional HS uses fixed, pre-defined parameters throughout the entire search process, making it hard to determine a good setting without a good amount of trial runs. The search results give no hint on which parameter should be adjusted in order to gain better performance. However, simple statistics gathered from the search process can yield useful information:

- Best Score (BS) indicates the fitness function evaluation of the search result, a search that finds better solution is deemed more favourable.
- Update Count (UC) is the number of better harmonies being included during the iterations. A higher UC shows a more successful improvisation process, with the overall quality of the harmonies stored in the harmony memory being improved more frequently.
  Note that in HS, if a newly improvised harmony evaluates worse than the ones stored in the harmony memory, it will be discarded completely. In another sense, the entire iteration is wasted, contributing nothing to the final solution.
- Last Best Iteration (LBI) is the iteration where the final optimal solution is found. A larger LBI indicates that the best solution was found later during the iteration process. This implies that a larger percentage of iterations were exploited to make contribution to the final solution.
  When performing a search, the number of iterations, i.e. the number of fitness function evaluations, is fixed based on the *maximum number of iterations* (K_{max}). The quality of candidate solutions has no impact on the run time of the search process. This typically means two things: 1) The optimal solutions are found well before K_{max}, and the remaining iterations contribute nothing to the final result and are therefore wasted. 2) The algorithm did not find a better solution within the maximum number of iterations and has been forced to terminate; with better pre-defined parameters, and more iterations, a better result can be achieved.

A number of deterministic parameter control rules are here proposed to fine-tune HS parameters dynamically, in order to address these issues raised due to fixed parameters. The aim is not only to produce a better search result at the end, but also to maximise the use of every iteration, i.e. higher UC, and larger LBI. The effect of these rules is demonstrated in comparisons with the original algorithm, using a minimisation problem fully described in [11]. The results were averaged across 100 runs under the same parameter settings. A relatively small $K_{max} = 2000$ is purposely chosen to demonstrate how much faster the dynamic approaches can converge.

A. Dynamic Harmony Memory Considering Rate

Harmony Memory Considering Rate (HMCR) is the core parameter of HS. It determines whether the new note value should be chosen from the harmony memory, or randomly selected from the range of all possible values. HS with a low HMCR takes less consideration of the historical values but focuses more on the entire value range when improvising new harmonies. HS with a high HMCR tries to produce a new harmony out of existing values stored in the harmony memory, and explores less outside. A dynamic HMCR that increases its value as the search progresses can be formulated such that

$$HMCR(K) = HMCR_{min} + \frac{HMCR_{max} - HMCR_{min}}{K_{max}} \times K$$

TABLE I
Impact of Dynamic HMCR

HMCR	Best Score	S.D.	Update Count	Last Best Iteration
0.5–1	7867.32	271.43	164.78	1671.86
0.5	9378.50	729.45	96.22	972.63
0.6–1	7887.69	328.40	166.43	1727.07
0.6	9416.34	713.45	97.72	1055.8
0.7–1	7870.75	269.39	163.88	1707.07
0.7	9358.83	764.44	95.17	982.74
0.8–1	7929.26	335.72	161.66	1682.02
0.8	9279.76	763.93	94.5	916.93
0.9–1	7887.82	302.40	162.34	1687.96
0.9	9341.67	694.85	97.45	1052.1

Table I shows the differences between the HS results using fixed and dynamic HMCR.

B. Dynamic Harmony Memory Size

Harmony Memory Size (HMS) controls the maximum number of best solutions that can be stored during the search process. A small HMS gives each musician less choices when improvising a new harmony. In the extreme case, where $HMS = 1$, HS keeps the best solution and discards the rest, with the musicians picking up the same note every time, unless otherwise forced to change by HMCR. A large HMS on the other hand, gives the musicians more choices. The other extreme case involves an infinitely large harmony memory which stores all the possible values for each variable, making the improvisation process the same as generating a new random harmony.

At the beginning of a search, as the musicians just start exploring the solution space, they do not have many good solutions. A small HMCR may cause them to randomly select values outside the harmony memory. Therefore, a large harmony memory is not required. In most of such cases, having a large pool of indecent harmonies only confuses the musicians, preventing them from choosing good values during improvisation. As the search approaches the end, the musicians have found many sub-optimal harmonies. In such cases, given a high HMCR, they will almost exclusively choose values out of the harmony memory when improvising new harmonies. Thus, a large pool of good results may contribute to a better solution.

From the above observation, a good dynamic HMS can be defined as

TABLE II
IMPACT OF DYNAMIC HARMONY MEMORY SIZE

HMS	Best Score	S.D.	Update Count	Last Best Iteration
10–20	7625.04	229.61	165.12	1581.58
20	7646.35	281.04	196.82	1594.93
15–30	7658.61	241.14	213.33	1450.18
30	7730.75	241.02	250.44	1553.89
20–40	7740.16	247.81	252.92	1500.9
40	7747.32	228.10	294.83	1525.12
25–50	7754.72	243.29	290.2	1416.99
50	7864.44	259.76	336.11	1538.25
30–60	7815.98	289.56	326.98	1446.94
60	7900.25	280.82	374.86	1462.44

$HMS(K) = HMS_{min} + \frac{HMS_{max} - HMS_{min}}{K_{max}} \times K$

Table II illustrates the effect of having a dynamically sized harmony memory. For each pair of tests, the dynamic approach sets its maximum memory size $HMS_{max} = HMS_{fixed}$, and $HMS_{min} = \frac{1}{2}HMS_{max}$. For all pairs, the dynamic approach finds better averaged best scores. Despite having a lower total number of harmony memories throughout the search, the dynamic schema have comparable update counts and last best iterations. This indicates more iterations were exploited to make contribution to the final solution.

C. Dynamic Pitch Adjusting Rate and Scalable Bandwidth

Pitch Adjusting Rate (PAR) and Bandwidth (BW) are two important parameters which mostly affect the rate of finding and converging to the optimal solution. They show the greatest impact when solving continuous valued optimisation problems. Traditional HS uses pre-defined and fixed PAR and BW throughout the search. This either results in slow initial exploration in the solution space, or inefficiency in searching for the optimal solution. Dynamic PAR and BW adjustment methods have been proposed to address this issue [14]. This can be outlined as follows:

$$PAR(K) = PAR_{min} + \frac{PAR_{max} - PAR_{min}}{K_{max}} \times K$$

$$BW(K) = BW_{max} \times exp(\frac{ln(\frac{BW_{min}}{BW_{max}})}{K_{max}}) \times K$$

However, when assigning the BW values, neither the original HS nor IHS takes the value ranges of each variable into consideration. As such, an adjustment too small will make very little impact upon values with a large range. For example, shifting variable x from 100 to 100.01 given the possible range of x being {-1000, +1000} makes little change, whilst a 0.01 adjustment is rather significant for a variable with a value range of {0, 1}. In this paper, a modified BW which scales with the ranges of each variable is used. When doing pitch adjustment, the new value assignment formula is thus changed herein into:

$X_{new} = X_{old} \pm RANDOM() \times BW \times RANGE_X$
where

$$BW(K) = BW_{max} \times exp(\frac{ln(\frac{BW_{min}}{BW_{max}})}{K_{max}}) \times K$$
$0 \leq BW \leq 1, 0 \leq RANDOM() \leq 1.$

D. A Combined Approach

All aforementioned individual parameter adjustment strategies can be combined together for a greater performance gain, allowing different sets of parameter settings for different search stages, as summarised in Table III. After initialisation, the algorithm employs a large harmony memory, with a large chance of randomly selecting new values, a small chance of selecting neighbouring values, and a large neighbour distance bandwidth. Towards the intermediate stage, the algorithm uses a medium sized harmony memory, with a balanced possibility between choosing values from the harmony memory and the range of all possible values, and with more frequent pitch adjustment and a lower distance bandwidth. Finally, towards the termination of the process, the algorithm utilises a small harmony memory, with the values chosen almost purely from stored good solutions, and very frequent pitch adjustment with a tiny distance bandwidth. It is worth mentioning that these stages are listed here for purely conceptual reasons, with no clear boundaries in between, as the algorithm shifts from one stage to another gradually during the search.

TABLE III
PARAMETER SETTINGS IN DIFFERENT SEARCH STAGES

	Initialisation	Intermediate	Termination
HMCR	Small	Medium	Large
MS	Small	Medium	Large
PAR	Small	Medium	Large
BW	Large	Medium	Small
Effect	High Diversity More Exploration	Steady Improving Harmonies	Fine Tuning Fast Convergence

The combined approach offers a better exploration of the initial solution space, a steady improvement to the overall quality of harmonies throughout the search, and a fine tuning stage towards termination that allows better convergence to the optimal solution. This is confirmed by the following experimental evaluations.

IV. EXPERIMENTATION AND DISCUSSION

Several optimisation problems taken from the literature are used to show the performance of the combined parameter adjustment method, including: two unconstrained (IV-A, IV-B) and two constrained (IV-C, IV-D) mathematical function optimisations, and two constrained engineering optimisations (IV-E, IV-F).The experimentation results are then compared with the solution achieved using IHS and original HS, along with the optimal solution, if known. The parameters settings used for all the example problems are listed in Table IV.

TABLE IV

PARAMETERS USED

HMCR	HMS	PAR	Scalable BW	K_{max}
0.5 – 0.95	10 – 20	0.35 – 0.99	0.00001 – 0.01	50,000

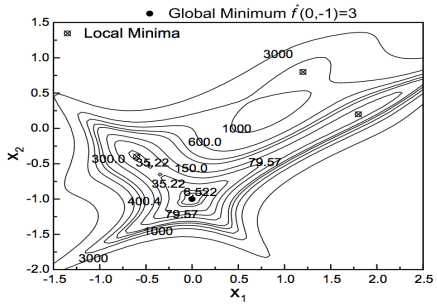

Fig. 2. Goldstein & Price Function I

A. Unconstrained Function: Goldstein & Price Function I

Minimise

$$f(\vec{x}) = \{\ 1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\ \} \times \{\ 30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\ \} \quad (1)$$

Search domain: $-2 \leq x_i \leq 2, i = 1, 2$.

TABLE V

GOLDSTEIN & PRICE I RESULT COMPARISON

Variables	Combined	IHS	HS	Optimal
x_1	0.000000	0.000000	-0.0000087289	0
x_2	-0.999999	-1.000001	-1.0000001192	-1
$f(\vec{x})$	3.000000	3.000000	3.000000000	3

This function is an eighth-order polynomial with two variables [10]. As shown in Fig. 2 (taken from [13]), the function has 4 local minima, one of which is the global best, $f(0, 1) = 3$. The best result achieved by the combined method found $f(x) = 3.0000000000032303$ which is less than 1×10^{-11} in terms of error with respect to the optimal value.

B. Unconstrained Function: Goldstein & Price Function II

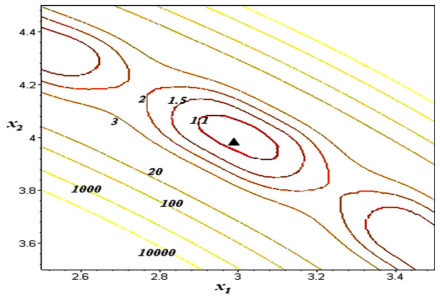

Fig. 3. Goldstein & Price Function II

Minimise

$$f(\vec{x}) = exp\{\ \frac{1}{2}(x_1^2 + x_2^2 - 25)^2\ \} + sin^4(4x_1 - 3x_2) + \frac{1}{2}(2x_1 + x_2 - 10) \quad (2)$$

TABLE VI

GOLDSTEIN & PRICE II RESULT COMPARISON

Variables	Combined	IHS	HS	Optimal
x_1	3.000002	3.00000	2.9998245239	3
x_2	3.999999	3.99999	4.0001201630	4
$f(\vec{x})$	1.000000	1.00000	1.000000000	1

This function [10] has many local minima. The global minimum is $f(3, 4) = 1$, with the function values illustrated in Fig.3 (taken from [14]). The search regions of the two variables are bounded between {-50, 50}.

C. Constrained Function: Disjoint Feasible Region

Maximise

$$f(\vec{x}) = \frac{100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2}{100} \quad (3)$$

This problem, originally discussed in [15], has three variables (x_1, x_2, x_3), one nonlinear inequality constraint $g(\vec{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0$, and bounded search domains $0 \leq x_i \leq 10, i = 1, 2, 3$. A solution is only feasible if and only if there exist p, q, r such that $p, q, r = 1, 2, ..., 9$ which jointly satisfy the inequality constraint. The optimal solution is $f(5, 5, 5) = 1$.

TABLE VII

DISJOINT FEASIBLE REGION RESULT COMPARISON

Variables	Combined	IHS	Coello	Koziel et al.
x_1	4.999510	5.000000	5.0000	N/A
x_2	4.999709	4.999999	5.0000	N/A
x_3	4.999629	5.000001	5.0000	N/A
$f(\vec{x})$	0.999999	0.9999999	1.000000	0.9999998

A variety of methods [14], [2], [12] have been used to solve this problem, their results are listed in Table VII.

D. Constrained Function II

Minimise

$$f(\vec{x}) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 \quad (4)$$

Subject to:

$$g_1(\vec{x}) = 4.84 - (x_1 - 0.05)^2 - (x_2 - 2.5)^2 \geq 0$$
$$g_2(\vec{x}) = x_1^2 + (x_2 - 2.5)^2 - 4.84 \geq 0$$
$$0 \leq x_i \leq 6, i = 1, 2$$

This minimisation problem has two variables, two inequality constraints and four boundary conditions. As illustrated in Fig. 4 (taken from [14]), the unconstrained global minimum is at $f(3, 2) = 0$. However, the added constraints make it no longer feasible. The actual feasible solutions only occupy approximately 0.7% of the total search space [14].

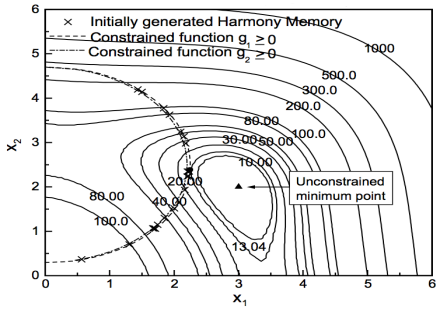

Fig. 4. Constraint Function II

TABLE VIII
CONSTRAINT FUNCTION II RESULT COMPARISON

Variables	Combined	IHS	HS	Deb
x_1	2.246827	2.2468258	2.246840	N/A
x_2	2.381877	2.381863	2.382136	N/A
$f(\vec{x})$	13.590842	13.590841	13.590845	13.59085
Constraints	All	None	All	All

This problem has been addressed by a number of techniques [14], [5], [9]. The result reported by IHS was found with no constraints active. The combined approach achieved $f(x) = 13.590841721017531$ with all constraints active.

E. Engineering Optimisation I: Spring Weight Minimisation

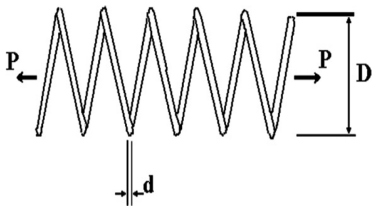

Fig. 5. Spring Weight Minimisation

This problem consists of minimising the weight of a tension spring subject to constraints on shear stress, surge frequency and minimum deflection [1], as shown in Fig. 5. The three variables are the mean coil diameter D, the wire diameter d, and the number of active coils N. The constraints are defined as follows:

Minimise

$$f(\vec{x}) = (x_3 + 2)x_2x_1^2 \tag{5}$$

Subject to

$$g_1(\vec{x}) = 1 - \frac{x_2^3x_3}{71785x_1^4} \leq 0,$$
$$g_2(\vec{x}) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leq 0,$$
$$g_3(\vec{x}) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0,$$
$$g_4(\vec{x}) = 1 - \frac{x_2 + x_1}{1.5} - 1 \leq 0,$$

The solutions found using different approaches are listed in Table IX. The combined method found a better solution of $f(\vec{x}) = 0.012672871994187039$ with all constraints active. The ever reported best result $f(\vec{x}) = 0.0126706$ [14] has a different evaluation of 0.0128874 in the experiment which actually violated the g_3 constraint ($g_3(\vec{x}) = 0.01367 \geq 0$), and is therefore not suitable for use in result comparison.

TABLE IX
WEIGHT OF SPRING RESULT COMPARISON

Variables	Combined	IHS	Belegundu	Coello
d	0.052310123	0.05115438	0.050000	0.051989
D	0.37183077	0.34987116	0.315900	0.363965
N	10.45541634	12.0764321	14.25000	10.890522
$f(\vec{x})$	0.012673	0.0126706	0.012833	0.012681
Constraints	All	None	All	All

F. Engineering Optimisation II: Welded Beam Design

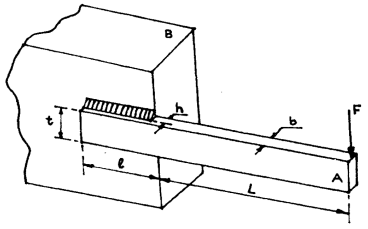

Fig. 6. The Welded Beam Structure

The welded beam [16] shown in Fig. 6 is a practical design problem often used as a benchmark for optimisation methods. The objective is to find the minimum fabricating cost of the welded beam subject to constraints on shear stress τ, bending stress σ, buckling load P_c, end deflection δ plus other side constraints. The four variables in the function represent the dimensions h, l, t and b of the welded beam assembly. The problem can be formulated as follows:

Minimise

$$f(\vec{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2) \tag{6}$$

Subject to

$$g_1(\vec{x}) = \tau(\vec{x}) - \tau_max \leq 0,$$
$$g_2(\vec{x}) = \sigma(\vec{x}) - \sigma_max \leq 0,$$
$$g_3(\vec{x}) = x_1 - x_4 \leq 0,$$
$$g_4(\vec{x}) = 0.10471x_1^2 + 0.04811x_3x_4(14 + x_2) - 5 \leq 0,$$
$$g_5(\vec{x}) = 0.125 - x_1 \leq 0,$$
$$g_6(\vec{x}) = \delta(\vec{x}) - \sigma_max \leq 0,$$
$$g_7(\vec{x}) = P - P_c(\vec{x}) \leq 0,$$

where

$$\tau(\vec{x}) = \sqrt{(\tau')^2 + \tau'\tau''\frac{x_2}{R} + (\tau'')^2}$$
$$\tau' = \frac{P}{\sqrt{2}x_1x_2}, \tau'' = \frac{MR}{J}, M = P(L + \frac{x_2}{2})$$

$$R = \sqrt{\frac{x_2^2}{4} + (\frac{x_1 + x_3}{2})^2}$$
$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + (\frac{x_1 + x_3}{2})\right]\right\}$$
$$\tau(\vec{x}) = \frac{6PL}{x_4x_3^2}, \delta(\vec{x}) = \frac{4PL^3}{Ex_3^3x_4}$$
$$P_c((\vec{x})) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}(1 - \frac{x_3}{L}\sqrt{\frac{E}{4G}})$$

$P = 6000lb, L = 14in, E = 30 \times 10^6 psi, G = 12 \times 10^6 psi$
$\tau_max = 13,600psi, \sigma_max = 30,000psi, \delta_max = 0.25in$

A number of different optimisation techniques [6], [4] have been used to solve this problem, with results shown in TableX. The combined method found the best result $f(\vec{x}) = 2.175095482060236$ with all constraints being active. The better results reported in [14], [4] were obtained with no constraints active (both violating g_1).

TABLE X
WELDED BEAM DESIGN RESULT COMPARISON

Variables	Combined	IHS	HS	Deb	Coello
h	0.21225574	0.20573	0.2442	0.2489	0.2088
l	8.04586947	3.47049	6.2231	6.1730	3.4205
t	7.88399070	9.03662	8.2915	8.1789	8.9975
b	0.21240242	0.20573	0.2443	0.2533	0.2100
$Cost$	2.175678	1.7248	2.3807	2.4328	1.7483
Constraints	All	None	All	All	None

V. CONCLUSION

This paper has discussed the impact of constant parameters in harmony search, and suggested several strategies for tuning them individually and dynamically. The improvements include better exploration of the solution space, maximised use of all iterations, and fine-tuning towards optimal solution. Dynamically tuned HS is good at locating global optimal region and producing optimal solution. A combined method that exploits all the individual parameter tuning strategies is also proposed. A number of optimisation benchmarks tests have been carried out on the combined method, including both unconstrained and constrained mathematical functions, and also real world engineering optimisation problems. Experimental results have demonstrated the performance of the combined method, especially when dealing with real world constrained optimisation problems, supported with comparative studies. A more comprehensive experimentation is still needed for a through analysis of the algorithm performance. Each parameter control rule needs to be studied further for a better understanding of their effects. Further statistical measures are also necessary to better justify the significance of these improvements.

Instead of using a predefined set of parameter adjustment rules, HS may perform even better if the parameters are dynamically adjusted at run time with respect to actual search performance, i.e. by adaptive parameter control. The adaptive HS will not only find optimised solution, but also learn the problem at hand as the search progresses. For example, if the UC did not increase, and no better solution was found for a large number of iterations, it might indicate that the algorithm had converged prematurely. As the optimal solution will always be preserved in the harmony memory, it is safe to restart the search and explore alternative regions of the solution space. This may be done by adjusting parameters significantly towards the original setting for initialisation, in order to ensure that the algorithm is not stuck at a local optimal. A better stopping criterion can also be adapted on the basis of UC and LBI, allowing an earlier termination if the algorithm has already converged. Alternatively, predefined K_{max} may be further extended if the search increases UC and refreshes LBI frequently even at final iterations (as the algorithm is still actively searching for the optimal solution). Work is on-going along these directions to improve the current research.

REFERENCES

[1] A.D. Belegundu, A Study of mathematical programming methods for structural optimization, PhD thesis, Department of Civil an Environmental Engineering, University of Iowa, 1982.

[2] C.A.C. Coello, Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, Advanced Engineering Informatics, vol. 16, pp. 193–203, 2002.

[3] C.A.C. Coello, Constraint-handling using an evolutionary multiobjective optimization technique, Civil Engineering and Environmental Systems, vol. 17, pp. 319–346, 2000.

[4] C.A.C. Coello, Use of a self-adaptive penalty approach for engineering optimization problem, Computers in Industry, vol. 41, no. 2, pp. 113–127, 2000.

[5] K. Deb, An efficient constraint handling method for genetic algorithms, Computer Methods in Applied Mechanics and Engineering, vol. 186, pp. 311–338, 2000.

[6] K. Deb, Optimal design of a welded beam via genetic algorithm, American Institute of Aeronautics and Astronautics Journal, vol. 29, no. 11, 1991.

[7] K. Deb and A.S. Gene, A robust optimal design technique for mechanical component design, Evolutionary Algorithms in Engineering Applications, Springer, Berlin, pp. 497–514, 1997.

[8] A.E. Eiben, R. Hinterding and Z. Michalewicz, Paramter Control in Evolutionary Algorithms, IEEE Transactions on Evolutionary Computation, vol. 3, no. 2, pp. 124–141, 1999.

[9] Z.W. Geem, J.H. Kim and G.V. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation, vol. 76, no. 2, pp. 60–68, 2001.

[10] A.A. Goldstein and J.F. Price, On descent from local minima, MATHEMATICS OF COMPUTATION, vol. 25, pp. 569–574, 1971.

[11] B.K. Kannan and S.N. Kramer, An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design, Journal of mechanical design, vol. 116, pp. 318–320, 1994.

[12] S. Koziel and Z. Michalewicz, Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization, Evolutionary Computation, vol. 7, no. 1, pp. 19–44, 1999.

[13] K.S. Lee and Z.W. Geem, A new meta-heuristic algorithm for continues engineering optimization: harmony search theory and practice, Computer Methods in Applied Mechanics and Engineering, vol. 194, pp. 3902–3933, 2004.

[14] M. Mahdavi, M. Fesanghary and E. Damangir, An improved harmony search algorithm for solving optimization problems, Applied Mathematics and Computation, vol. 188, pp. 1567–1579, 2007.

[15] Z. Michalewicz and M. Schoenauer, Evolutionary algorithms for constrained parameter optimization problems, Evolutionary Computing, vol. 4, no. 1, pp. 1–32, 1996.

[16] J.N. Siddall, Analytical Decision-Making in Engineering Design, Prentice-Hall, Englewood Cliffs, New Jersey, 1972.

[17] A. Vasebi, M. Fesanghary and S.M.T. Bathaeea, Combined heat and power economic dispatch by harmony search algorithm, International Journal on Electric Power, vol. 29, pp. 713–719, 2007.